Configuring Nomad for High Availability

Eric Wright
Technology Evangelist, Podcaster

@DiscoPosse www.discoposse.com www.discopossepodcast.com

Overview

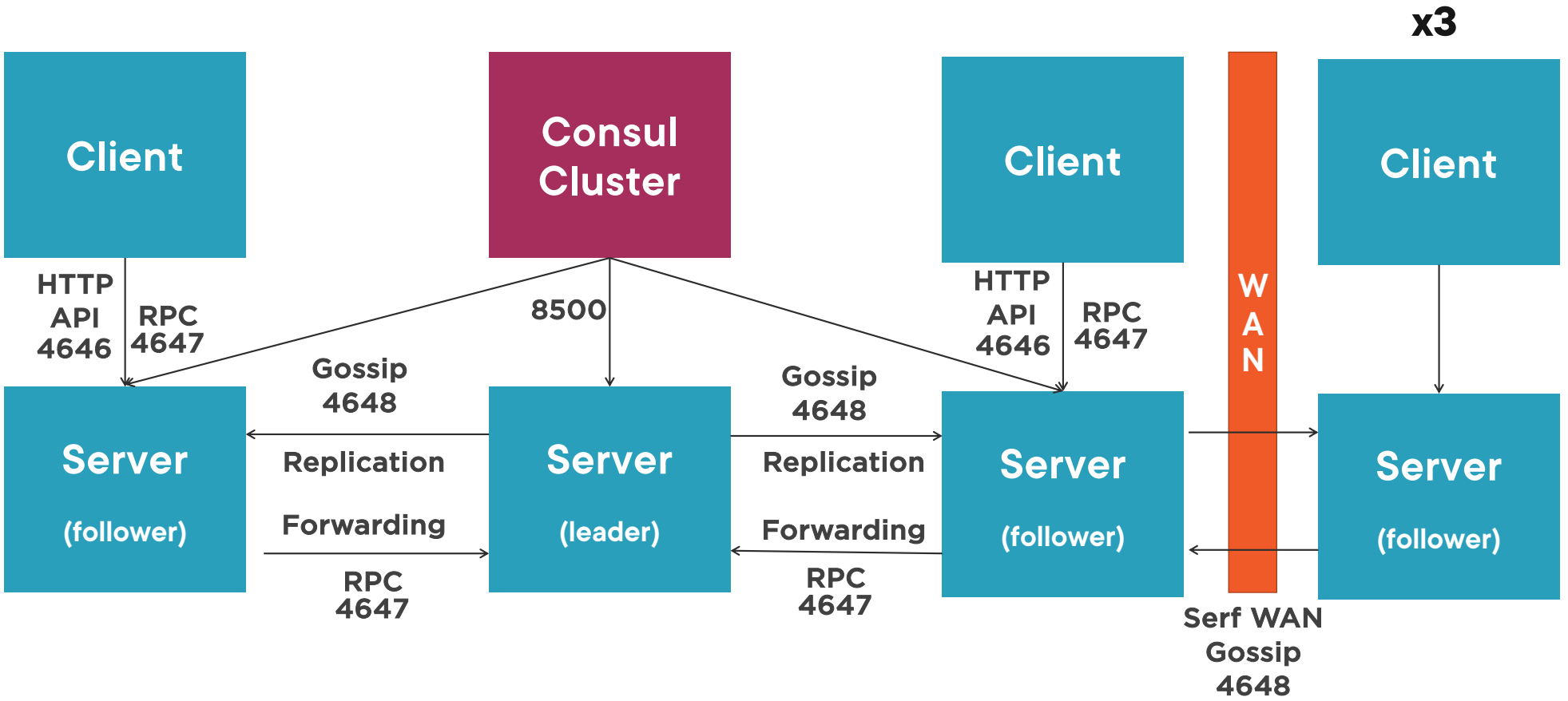
- Explore failure domains in a Nomad environment
- Review the Nomad architecture for resiliency options
- Build a resilient job specification
- Examine job and cluster behavior during a failure scenario
- Configure Autopilot for Nomad

A Primer on Nomad Failure Domains

- Global failure risks can include:
  - Security and permissions
  - Firewalls and network connectivity
  - Service registry

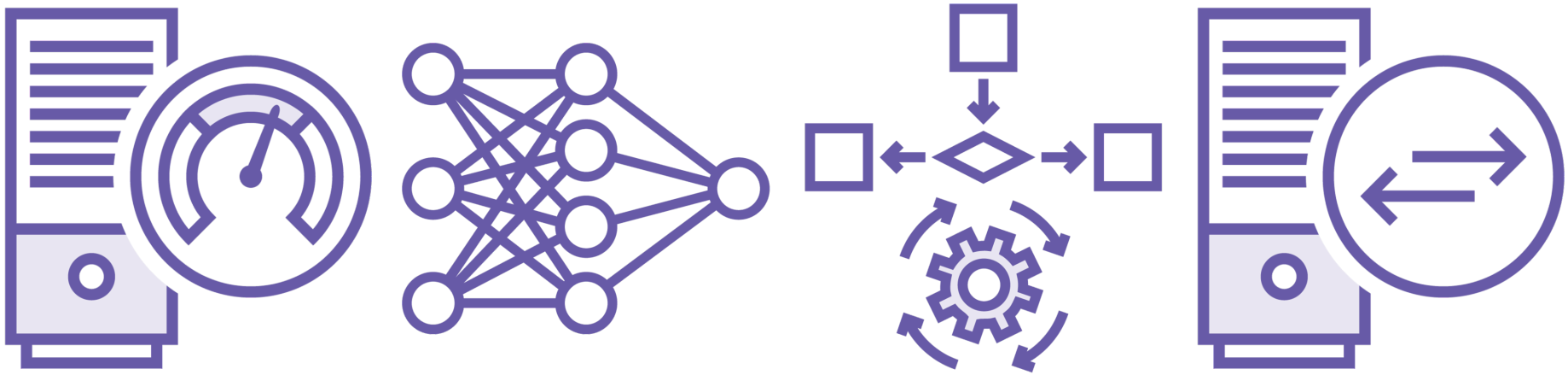
- Node
  - Driver failure – e.g. Docker
  - Nomad agent failure
  - Job/task failure
  - Network connectivity
- Cluster
  - Cluster quorum
  - Network connectivity
  - Resource capacity
- Region
  - Network connectivity
  - Resource capacity

Resiliency by Design



Resiliency Options in Nomad

Node Resiliency



Monitor Hardware

Monitor Services

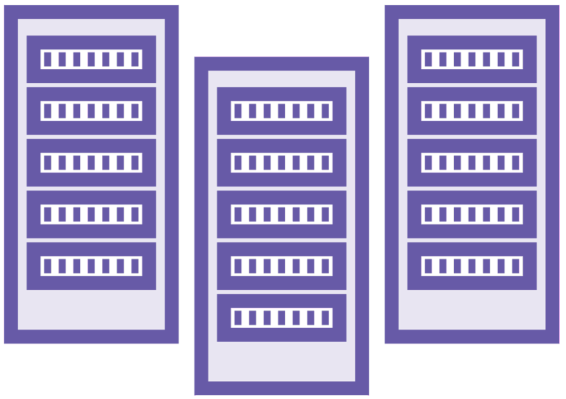
Automate Deployment

Automate Remediation

Cluster and Region Resiliency

Monitor cluster and region health

Automate remediation and recovery



Scale for resiliency and capacity

- Leverage Consul for service discovery
- Create immutable deployment processes
- Deploy full-stack monitoring
- Automate service recovery
- Use SRE operations methods

Designing Jobs for Resiliency

```
# Specify the datacenters within the region this job can run in.
# Must be provided and does not have a default setting
datacenters = ["dc-aws-1", “dc-aws-2”]

# Specify this job to have rolling updates, two-at-a-time, with 30 second intervals
update {
    stagger      = "30s"
    max_parallel = 2
  }

service {
      # This tells Consul to monitor the service on the port labelled "http”
      port = "http"

      check {
        type     = "http"
        path     = "/health"
        interval = "10s"
        timeout  = "2s"
      }
    }
```

Distribute Jobs and Tasks in the Job Specification

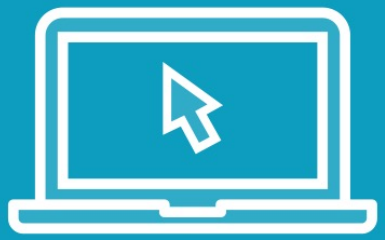
Distribute the jobs and tasks using multiple datacenters as well as other capabilities like deployments, rolling updates, and health checks.

Using the Spread Stanza

myjob.nomad

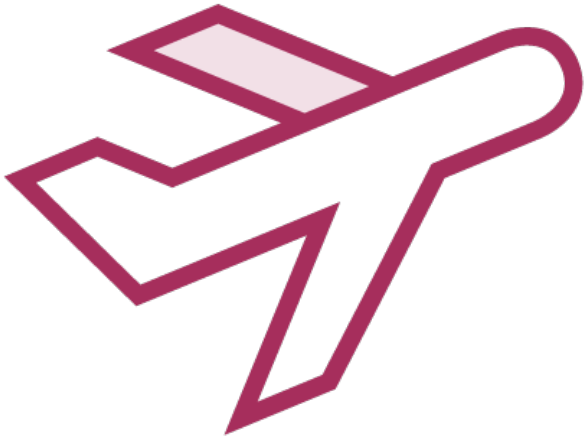
```
job ”myjob" {
     datacenters = [“dc-aws-1”, “dc-aws-2”]
     # Spread allocations over all datacenter
     spread {
          attribute =  "${node.datacenter}"
     }
     group "test" {
          count = 10
          # Spread allocations over each rack
          spread {
               attribute =  "${meta.rack}"
               target "r1" {
                    percent =  60
               }
               target "r2" {
                    percent =  40
               }
          }
     }
}
```

Demo



- Force failure of different resources
- Observe system behaviors during:
  - Node and service outages
  - Recovery of service

Nomad on Autopilot



Autopilot is operator-friendly

Configured in the server stanza on all members

Used to monitor raft

Safely clean up dead servers

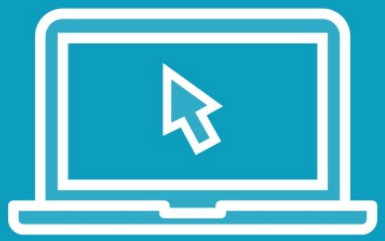
Safe introduction of new servers

Autopilot configuration

server.hcl

```
autopilot {
  cleanup_dead_servers      = true
  last_contact_threshold    = "200ms"
  max_trailing_logs         = 250
  server_stabilization_time = "10s"
  enable_redundancy_zones   = false
  disable_upgrade_migration = false
  enable_custom_upgrades    = false
}
```

Demo



- **Review the Autopilot configuration**
- **Update Autopilot using Nomad operator**

Summary

- Reviewed failure domains in a Nomad environment
- Evaluated the Nomad architecture for commonly used resiliency options
- Created a real failure scenario to test job and cluster behavior

Up Next:
Performing Data and Configuration Backups